DS Bioperl cours

Utiliser Perl et la bibliothèque Bioperl pour chacune des questions. Les exercices sont indépendants les uns des autres. Je ne tiendrai pas compte de l'utilisation des « use » (donc ni des « my ») dans les scripts perl.

1) Quel est l'intérêt de Bioperl par rapport à du Perl ?
Estimation : 1 minute - 0,25 point

2) Convertisser en Bioperl le script suivant : Estimation : 7 minutes – 1,75 points

```
use strict;
use warnings;
my $seq=$ARGV[0];
chomp($seq);
my @split=split("",$seq);
my $chaine="";
foreach (@split){
»	$chaine=$_.$chaine;
}
if ($seq=~m/^(A|T|G|C)*$/){»
»	print "ADN\n".$seq."\n";
»	print $chaine;
}
elsif ($seq=~m/^(A|U|G|C)*$/){»
»	print "ARN\n".$seq."\n";
»	print $chaine;
}
else{
»	print "Protein\n".$seq."\n";»
}
```

3) Convertisser en Bioperl le script suivant : Estimation : 7 minutes – 1,75 points
N'écrivez pas la séquence (ATG***) en entière

```
use strict;
use warnings;
my $seq="TGTGTGTGAATTCATGCATATGTGAATTCTATGCATGCAA";
my $coupe="GAATTC";
my $coupe_avant="G";
my $coupe_apres="AATTC";
my $enzyme="EcoRI";
my @fragments=split($coupe,$seq);
if ($#fragments>=1){
»	my $premier=shift(@fragments);
»	my $dernier=pop(@fragments);
»	print $premier.$coupe_avant."\n";
»	foreach (@fragments){
»	»	print $coupe_apres.$_.$coupe_avant."\n";
»	}
»	print $coupe_apres.$dernier."\n";
}
else{
»	print $seq."\n";
}
```

4) Convertisser en Bioperl le script suivant : Estimation : 7 minutes – 1,75 points
N'écrivez pas la séquence (ATG***) en entière

```
use strict;
use warnings;
my $sequence="";
my $nom_sequence="";
open (F,"sequences.fa");
my $bool=0;
while (<F>){
»	my $ligne=$_;
»	chomp($ligne);
»	if ($ligne=~m/^>/){
»	»	$bool=0;»
»	}
»	if ($ligne=~m/ENST00000512366.1/){
»	»	$bool=1;
»	}
»	if ($bool==1){
»	»	print $ligne;»
»	}
}
close(F);
```

5) Lire le fichier blast « seq.bls » et afficher les séquences requêtes qui possèdent un hit dont le nom contient « deshydrogenase ». Précision : une séquence requête doit être affichée qu'une seule fois même si elle a plusieurs hits « deshydrogenase ». Estimation : 12 minutes - 2,5 points

6) Pour une séquence contenue dans le fichier fasta « file.fasta » compter le nombre de mots de taille 1 jusqu'à 10. Estimation : 12 minutes - 2,5 points
Affichage voulue :
Mot de taille : 1
G 225
C 260
A 342
Mot de taille : 2
...
Mot de taille : 10
GGGCACCTTC 1
...

7) Créer une séquence puis la traduire une séquence en fonction des paramètres envoyés dans le script : le premier paramètre va contenir : « + » ou « - » afin de choisir le sens du brin (sens ou antisens), le second paramètre va contenir : 1,2 ou 3 afin de choisir la cadre de lecture.
Estimation : 12 minutes - 2,5 points

8) Corriger les erreurs dans ce script (les use sont tous bons) : Écrire un programme qui permet de convertir plusieurs fichiers au format Genbank dans un seul fichier fasta. Ce script doit lire un fichier « liste.txt » contenant les noms des fichiers genbank à convertir. Un fichier genbank contient une seule séquence fasta. Le nom de fichier fasta doit avoir le prefix « short » ou « long » en fonction de la taille de la séquence : « short_leptine.fasta », « long_sequence.fasta ». « short » veut dire que la séquence à une taille inférieure à la taille moyenne de toutes les séquences, « long » que la séquence à une taille supérieure ou égale à la taille moyenne de toutes les séquences.
Estimation : 16 minutes - 3,5 points

« liste.txt » contient un nom de fichier genbank par ligne :
sequence.gb
leptine.gb
Script :

```
use strict;
use warnings;
use Bio::SeqIO;



open (F,"liste.txt");
while (<F>){
»    my $file=$_;
»    chomp($file);
»    my $in=Bio::SeqIO->new(-file=>$file.".gb",-format=>"genbank");




}
my $moyenne=0;
open (F,"liste.txt");
while (<F>){
»    my $file=$_;
»    chomp($file);
»    my $in=Bio::SeqIO->new(-file=>$file.".gb",-format=>"genbank");
»    my $prefix="long";
»    my $seq  = $in->next_seq;
»    if ($seq<$moyenne){
»    »    $prefix="short";
»    }
»    my $out=Bio::SeqIO->new(-file=>"".$prefix."_".$file.".fa",-format=>"fasta");
»    $out->write_seq($seq);
}
```

9) Corriger les erreurs dans ce script (les use sont tous bons) :
Estimation : 16 minutes - 3,5 points

Objectif du script

Afin de rechercher les similarités avec l'outil BLAST entre un fichier contenant une seule séquence (pour que ça soit plus simple) en format fasta et une base de données, le terminal doit afficher chaque nom des fichiers fasta contenue dans le dossier « seq » pour que l'utilisateur puisse choisir le fichier de son choix en entant le chiffre correspondant au fichier. Si l'utilisateur fait un choix impossible il faut afficher « Choix impossible » et reproposer les choix (rappel perl : il faut faire une boucle). Par exemple :
Choisissez un fichier séquence :
1)seq_1.fa
2)seq_2.fa
[L'utilisateur doit taper 1 ou 2 puis « entrer »]
#Si il tape 4 :
Choix impossible
Choisissez un fichier séquence :
1)seq_1.fa
2)seq_2.fa
[L'utilisateur doit taper 1 ou 2 puis « entrer »]
Puis, lancer la comparaison de type « blastn » entre le fichier contenant une séquence sélectionnée et la base de données « human ». Le fichier résultat doit porter le nom de la séquence requête (query) suivi de « _ » suivi du nom de la base de données utilisée suivi de « .bls » (exemple : « unknown1_human.bls »).
En lisant les résultats de blast en Bioperl (le format par défaut d'un fichier résultat de blastn est « blast »), annoter la séquence requête grâce au/aux HSP de son premier hit avec les paramètres suivants : primary_id « HIT », position start et la position end du/des HSP, le nom du hit en tag « name ». Enfin, imprimer la séquence annotée dans une fiche Genbank portant le nom de la séquence requête (query) suivi de « _ » suivi du nom de la base de données utilisée suivi de « .gb » (exemple : « unknown1_human.gb »).

```
use strict;
use warnings;
use Bio::Tools::Run::StandAloneBlastPlus;
my @lis=`ls seq`;#permet de récupérer les noms des fichier du répertoire « seq » dans le
#tableau @lis
my %fich_sequences;
my $i=1;
foreach (@lis){
	chomp($_);
	$fich_sequences{$i}=$_;
	$i++;
}
print "Choisissez un fichier de séquence :\n";
my $stdin_seq;
while (1){
	foreach (sort keys %fich_sequences){
	
	}
	$stdin_seq=<STDIN>;
	chomp($stdin_seq);
	if (defined($fich_sequences{$stdin_seq})){#defined() renvoi vrai si la clé existe
#dans le tableau associatif
		last; #permet de sortir de la boucle
	}


}
my $infasta=Bio::SeqIO->new('seq/'.$fich_sequences{$stdin_seq});
my $sequence_query=Bio::SeqIO->new(-seq =>'TATA');
my $nom="human";
my $fac=Bio::Tools::Run::StandAloneBlastPlus -> new(-db_name =>$nom);
$fac -> blastn(-query => "seq/".$fich_sequences{$stdin_seq},
outfile => $sequence_query."_".$nom.".bls");
$fac -> cleanup;
my $in = Bio::SearchIO->new(-format => 'blast', -file => $sequence_query->display_id."_".$nom.".bls");
my $result=$in->next_hit;
my $hit=$result->next_hsp ;
my $out=Bio::SeqIO->new(-file => '>'.$result->query_name."_".$nom.".gb",-format=> "GenBank");
while (my $hsp=$hit->next_result){
	my $feat=new Bio::SeqFeature::Generic();
	$feat->add_tag_value('name',$hit->name);
	$feat->start($hsp->start);
	$feat->end($hsp->end);
	$feat->primary_tag("HIT");

}
$out->write_seq($sequence_query);
```